REhome

Release 0.01

Katrin and Lukas

Oct 15, 2021

CONTENTS:

CHAPTER

ONE

INTRODUCTION

Welcome to the REhome documentation.

This section should describe the general functionality of the Webapp

1.1 Overview

Where to find what

1.2 How to contribute?

Write some contribution guildelines here.

CHAPTER

TWO

BUILDING MODULE

The Building module is used to calculate the heating and cooling demand of a building. It is separated into the following sub-packages.

2.1 Geometry

The geometry module is used to create and manipulate the geometry of a building. Output should consist of all opaque and transparent areas, their types (roof, outside_wall, window) and their orientations.

`building.geometry.`**`area_from_geopolygone`**(*polygone*)
  Calculate the area of a geopolygone.

`building.geometry.`**`facade_area`**(*perimeter*, *height*)

`building.geometry.`**`height_from_story`**(*n_story*, *story_height*)

`building.geometry.`**`perimeter_from_geopolygone`**(*polygone*)
  Calculate the perimeter of a geopolygone.

2.2 Location

`building.location.`**`read_location_data`**()
  load the location data from a csv file.

2.3 Physics

`building.physics.`**`heatDemand`**(*gains=[]*, *losses=[]*)
  Calculate the heating or cooling demand using an energy balance.

Parameters

- **gains** (*list of float*) – All heat gains of the building. Q_{gain} $[Wh]$
- **losses** (*list of float*) – All heat losses of the building. Q_{loss} $[Wh]$

Returns Heating (+) or cooling (-) demand. Q $[Wh]$

Return type float

Notes

$$Q = \sum Q_{loss} - \sum Q_{gain}$$

`building.physics.heatFlows(building, weatherdata)`

This function calculates the the different heatflows in the building

Parameters

- **building** (*dict*) – dictionary containing building parameters created with buildingFactory (building.yaml)
- **weatherdata** (*pandas dataframe*) – Dataframe containing weatherdata, can be created with utilities.read_tmy_data()

Returns Dataframe containing the following heatflows and heatdemand df['Qflow_int'] Internal gains $\dot{Q}_{int}$ $[W]df['Qflow_sol']Solargains : math$:dot{Q_{sol}}~[W] df['Qflow_vent'] Ventilation losses $\dot{Q}_{vent}$ $[W]df['Qflow_trans_facade']Transmissionlossesfacade : math$:dot{Q_{trans,facade}}~[W] df['Qflow_trans_roof'] Transmission losses roof $\dot{Q}_{trans,roof}$ $[W]df['Qflow_trans_ground']Transmissionlossesground : math$:dot{Q_{trans,ground}}~[W] df['Qflow_trans_windows'] Transmission losses windows :math:`dot{Q_{trans,windows}}~[W] df['heatDemand'] Heat demand :math:`Q_{th}~[Wh]

Return type pandas dataframe

`building.physics.heatflow2Energy(heatflow, timestep)`

Calculate the resulting energy of a heatflow in a certain timestep.

Parameters

- **heatflow** (*float*) – Heatflow. $\dot{Q}$ $[W]$
- **timestep** (*float*) – Timestep. Δt $[h]$

Returns Energy. Q $[Wh]$

Return type float

Notes

$$Q = \dot{Q} \cdot \Delta t$$

`building.physics.infAndVent(n, volume, tempIn, tempAmb)`

Calculate the infiltration and/or ventilation losses of a volume.

Parameters

- **n** (*float*) – Ventilation/Infiltration rate n $[1/h]$
- **volume** (*float*) – Volume of the construction element V $[m^3]$
- **tempIn** (*float*) – Temperature inside of the volume T_{in} $[C]$
- **tempAmb** (*float*) – Temperature outside of the volume T_{amb} $[C]$

Returns Heatflow through the construction element $\dot{Q}$ $[W]$

Return type float

Notes

$$\dot{Q} = n \cdot V \cdot \rho \cdot c_{P,air} \cdot (T_{in} - T_{amb})$$

`building.physics.internalGains(area, specInternalGains)`
Calculate the internal gains for the heated area.

Parameters

- **area** (*float*) – Heated living area A $[m^2]$
- **specInternalGains** (*float*) – Specific internal gains q_{int} $[W/m^2]$

Returns Heatflow of internal gains $\dot{Q}_{int}$ $[W]$

Return type float

Notes

$$\dot{Q}_{int} = q_{int} \cdot A$$

`building.physics.solarGains(gValue, area, irrad)`
Calculate the solar gains through a transparent plane.

Parameters

- **gValue** (*float*) – Solar heat gain coefficient g $[-]$
- **area** (*float*) – Area of the plane A $[m^2]$
- **irrad** (*float*) – Global irradiation perpendicular to the plane G_n $[W/m^2]$

Returns Heatflow through the plane $\dot{Q}$ $[W]$

Return type float

Notes

$$\dot{Q} = g \cdot A \cdot G_n$$

`building.physics.transmission(uValue, area, tempIn, tempAmb)`
Calculate transmission losses through a plane.

Parameters

- **uValue** (*float*) – Heat transfer coefficient U $[W/m^2K]$
- **area** (*float*) – Area of the plane A $[m^2]$
- **tempIn** (*float*) – Temperature inside of the plane T_{in} $[C]$
- **tempAmb** (*float or series*) – Temperature outside of the plane T_{amb} $[C]$

Returns Heatflow through the plane $\dot{Q}$ $[W]$

Return type float

Notes

$$\dot{Q} = U \cdot A \cdot (T_{in} - T_{amb})$$

2.4 Utilities

`building.utilities.get_tmy_data`(*latitude*, *longitude*)
Load weather - typical meterological year(TMY) data from PVGIS

Parameters

- **latitude** (*float*) – Latitude [decimal degrees]
- **longitude** (*float*) – Longitude [decimal degrees]

Returns

Return type content of the requests.response object containing weather data

`building.utilities.read_tmy_data`(*userID='userID'*, *filepath=PosixPath('.')*)
Read the weather data from a json file and write to pandas dataframe.

Parameters

- **userID** (*str*) – ID of the user
- **filepath** (*pathlib Path*) – filepath where the json file is saved

Returns

- *pandas df containing*
- **- time (index, datetime)** (*Date & time (UTC)*)
- **- T2m (pandas column, float)** (*Dry bulb (air) temperature [°C]*)
- **- RH (pandas column, float)** (*Relative Humidity [%]*)
- **- G(h) (pandas column, float)** (*Global horizontal irradiance [W/m2]*)
- **- Gb(n) (pandas column, float)** (*Direct (beam) irradiance [W/m2]*)
- **- Gd(h) (pandas column, float)** (*Diffuse horizontal irradiance [W/m2]*)
- **- IR(h) (pandas column, float)** (*Infrared radiation downwards [W/m2]*)
- **- WS10m (pandas column, float)** (*Windspeed [m/s]*)
- **- WD10m (pandas column, float)** (*Wind direction [°]*)
- **- SP (pandas column, float)** (*Surface (air) pressure [Pa]*)

Notes

A typical meteorological year (TMY) is a set of meteorological data with data values for every hour in a year for a given geographical location. The data are selected from hourly data in a longer time period (normally 10 years or more). The TMY is generated in PVGIS following the procedure described in ISO 15927-4.[1]

References

`building.utilities.save_tmy_data`(*data*, *userID='userID'*, *filepath=PosixPath('.')*)

Save the weather data in a json file.

Parameters

- **data** (*content of the requests.response object*) – content of the requests.response object containing weather data
- **userID** (*str*) – ID of the user
- **filepath** (*pathlib Path*) – filepath where the json file should be saved

[1] https://ec.europa.eu/jrc/en/PVGIS/tools/tmy

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX